Notice of Allowability	Application No.	Applicant(s)	
	09/915,509	BATES ET AL.	
	Examiner	Art Unit	
	INSUN KANG	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *3/29/2006*.

2. ☒ The allowed claim(s) is/are *30-33, 36-42, 45-51, 54, and 55 (renumbered as 1-20)* .

3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date *20090408* .
7. ☒ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Mr. Stewart (reg. 54,945) on 4/6-4/2008.

The application has been amended as follows:

1-29. (Canceled)

30. (Currently Amended) A computer-implemented method for debugging a program in a distributed debugging environment linking at least a first and second computing system over a communications network, comprising:

displaying, on a display interface of a debugger application executing on the first computing system, a value of one or more data variables of the program being debugged;

receiving a command to execute the program being debugged, wherein the program is executed by at least the second computing system;

in response to receiving the command, executing, by at least the second computing system, the program from a first execution point to a second execution point:

determining, based on the first and second execution points, which data variables are capable of being ~~may have been~~ modified by the execution of the program, wherein the determining comprises:

(i) if a node of a control flow graph that contains the first execution point of the program contains a breakpoint positioned after the first execution point, marking all data variables referenced during the execution of each statement of the node from the first execution point to the breakpoint as variables capable of being modified by the program during the execution of the program from the first execution point to the breakpoint, and

executing each statement of the node from the first execution point to the breakpoint;

(ii) otherwise, beginning with a root node, marking each node of the control flow graph from which flow of the program can be exited and reentered during execution of the program from the first execution point to the second execution point,

for each marked node, marking all data variables of the node as variables capable of being modified by the program during the execution of the program from the first execution point to the second execution point, and

beginning with the node containing the first execution point, generating a list of unmarked nodes reachable during execution of the program from the first execution point; and

refreshing the value displayed on the display interface of the debugger application executing on the first computing system, only for the data variables marked as capable of being ~~that may have been~~ modified by the execution of the program from the first execution point to the second execution point.

31. (Previously Presented) The method of claim 30, wherein the command comprises a step command configured to cause the execution of a statement of the program being debugged present at the first execution point.

32. (Currently Amended) The method of claim 31, wherein executing the program to the second execution point, comprises:

executing the statement; and

determining a set of variables ~~that may have been~~ modified by the execution of the statement.

33. (Previously Presented) The method of claim 30, wherein the command comprises a run command, and wherein the second execution point comprises a breakpoint encountered by the execution of the program from the first execution point.

34. (Cancelled)

35. (Cancelled)

36. (Currently Amended) The method of claim [[35]] 30, wherein generating the list comprises:

traversing the control flow graph from the node containing the first execution point of the program to each subsequent node by following program control flow defined by arcs;

adding each encountered node to the list if the encountered node is not marked and is not already in the list;

determining whether the encountered node contains a breakpoint; and

if so, terminating a traversal along a current traversal path.

37. (Currently Amended) The method of claim [[35]] 30, further comprising:

propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all data variables associated with all statements of the preceding unmarked node and all propagated kill variables propagated to the preceding unmarked node.

38. (Previously Presented) The method of claim 30, wherein the debugger interface is displayed on a first computer system, and wherein the program being debugged is executing on a second computer system, and wherein the first and second computer systems are configured to communicate over a network communication channel.

39. (Currently Amended) A computer-readable storage medium containing a program which, when executed, performs an operation for debugging in a distributed debugging environment linking at least a first and second computing system over a communications network, comprising:

displaying, on a display interface of a debugger application executing on the first computing system, a value of one or more data variables of the program being debugged;

receiving a command to execute the program being debugged, wherein the program is executed by at least the second computing system;

in response to receiving the command, executing, by at least the second computing system, the program from a first execution point to a second execution point:

determining, based on the first and second execution points, which data variables are capable of being ~~may have been modified~~ by the execution of the program, wherein the determining comprises:

(i) if a node of a control flow graph that contains the first execution point of the program contains a breakpoint positioned after the first execution point, marking all data variables referenced during the execution of each statement of the node from the first execution point to the breakpoint as variables capable of being modified by the program during the execution of the program from the first execution point to the breakpoint, and

executing each statement of the node from the first execution point to the breakpoint;

(ii) otherwise, beginning with a root node, marking each node of the control flow graph from which flow of the program can be exited and reentered during execution of the program from the first execution point to the second execution point,

for each marked node, marking all data variables of the node as variables capable of being modified by the program during the execution of the program from the first execution point to the second execution point, and

beginning with the node containing the first execution point, generating a list of unmarked nodes reachable during execution of the program from the first execution point; and

refreshing the value displayed on the display interface of the debugger application executing on the first computing system, only for the data variables marked as capable of being

~~that may have been~~ modified by the execution of the program from the first execution point to the second execution point.

40. (Currently Amended) The computer-readable storage medium of claim 39, wherein the command comprises a step command configured to cause the execution of a code statement of the program being debugged present at the first execution point.

41. (Currently amended) The computer-readable storage medium of claim 40, wherein executing the program being debugged to the second execution point, comprises:

executing the statement; and

determining a set of variables ~~that may have been~~ modified by the execution of the statement.

42. (Currently Amended) The computer-readable storage medium of claim 39, wherein the command comprises a run command, and wherein the second execution point comprises a breakpoint encountered by the execution of the program from the first execution point.

43. (Cancelled)

44. (Cancelled)

45. (Currently Amended) The computer-readable storage medium of claim [[43]] 39, wherein generating the list comprises:

traversing the control flow graph from the node containing the first execution point of the program to each subsequent node by following program control flow defined by arcs;

adding each encountered node to the list if the encountered node is not marked and is not already in the list;

determining whether the encountered node contains a breakpoint; and

if so, terminating a traversal along a current traversal path.

46. (Currently Amended) The computer-readable storage medium of claim [[43]], 39, wherein the operation further comprises:

propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all data variables associated with all statements of the preceding unmarked node and all propagated kill variables propagated to the preceding unmarked node.

47. (Currently Amended) The computer-readable storage medium of claim 39, wherein the debugger interface is displayed on a first computer system, and wherein the program being debugged is executing on a second computer system, and wherein the first and second computer systems are configured to communicate over a network communication channel.

48. (Currently Amended) A first computing device, comprising:

a processor;

a memory; and

a debugger program which when executed on the processor, performs a method for debugging an application executing on a second computing device in a distributed debugging environment, comprising:

displaying, on a display interface of a debugger application executing on the first computing system, a value of one or more data variables of the program being debugged;

receiving a command to execute the program being debugged, wherein the program is executed by at least the second computing system;

in response to receiving the command, executing, by at least the second computing system, the program from a first execution point to a second execution point:

determining, based on the first and second execution points, which data variables are capable of being ~~may have been modified~~ by the execution of the program, wherein the determining comprises:

(i) if a node of a control flow graph that contains the first execution point of the program contains a breakpoint positioned after the first execution point, marking all data variables referenced during the execution of each statement of the node from the first execution point to the breakpoint as variables capable of being modified by the program during the execution of the program from the first execution point to the breakpoint, and

executing each statement of the node from the first execution point to the breakpoint;

(ii) otherwise, beginning with a root node, marking each node of the control flow graph from which flow of the program can be exited and reentered during execution of the program from the first execution point to the second execution point,

for each marked node, marking all data variables of the node as variables capable of being modified by the program during the execution of the program from the first execution point to the second execution point, and

beginning with the node containing the first execution point, generating a list of unmarked nodes reachable during execution of the program from the first execution point; and

refreshing the value displayed on the display interface of the debugger application executing on the first computing system, only for the data variables marked as capable of being ~~that may have been~~ modified by the execution of the program from the first execution point to the second execution point.

49. (Previously Presented) The computing device of claim 48, wherein the command comprises a step command configured to cause the execution of a code statement of the program being debugged present at the first execution point.

50. (Currently Amended) The computing device of claim 49, wherein executing the program being debugged to the second execution point, comprises:

executing the statement; and

determining a set of variables ~~that may have been~~ modified by the execution of the statement.

51. (Previously Presented) The computing device of claim 48, wherein the command comprises a run command, and wherein the second execution point comprises a breakpoint encountered by the execution of the program from the first execution point.

52. (Cancelled)

53. (Cancelled)

54. (Currently Amended) The computing device of claim [[53]] 48, wherein generating the list comprises:

traversing the control flow graph from the node containing the first execution point of the program to each subsequent node by following program control flow defined by arcs;

adding each encountered node to the list if the encountered node is not marked and is not already in the list;

determining whether the encountered node contains a breakpoint; and

if so, terminating a traversal along a current traversal path.

55. (Currently Amended) The computing devices of claim [[53]] 48, wherein the operations for the program further comprise:

propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all data variables associated with all statements of the preceding unmarked node and all propagated kill variables propagated to the preceding unmarked node.

Drawings

2. Drawings filed on 7/26/2001 have been accepted.

Examiner's Statement of Reason(s) for Allowance

3. Claims 30-33, 36-42, 45-51, 54, and 55 (renumbered as 1-20) are allowed.

4. The following is an examiner's statement of reason s for allowance:

The prior arts of record, taken alone or in combination, fail to teach or fairly suggest at least:

if a node of a control flow graph that contains the first execution point of the program contains a breakpoint positioned after the first execution point, marking all data variables referenced during the execution of each statement of the node from the first execution point to the breakpoint as variables capable of being modified by the program during the execution of the program from the first execution point to the breakpoint, and executing each statement of the node from the first execution point to the breakpoint; otherwise, beginning with a root node, marking each node of the control flow graph from which flow of the program can be exited and reentered during execution of the program from the first execution point to the second execution point, for each marked node, marking all data variables of the node as variables capable of being modified by the program during the execution of the program from the first execution point to the second execution point, and beginning with the node containing the first execution point, generating a list of unmarked nodes reachable during execution of the program from the first execution point as recited in the independent claims 30, 39, and 48.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

5. Any inquiry concerning this communication or earlier communications from the examiner should be directed to INSUN KANG whose telephone number is (571)272-3724. The examiner can normally be reached on M-R 7:30-6 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis A. Bullock, Jr. can be reached on 571-272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Information regarding the status of an application may be obtained from the Patent Application Information

Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Insun Kang/
Examiner, Art Unit 2193